Olivia Smith 505584075

1. One big hurdle that I overcame was trying to figure out how all of the different pieces of the skeleton interacted with each other and understanding the function of each class and how to call the classes within a separate class. In addition, it was difficult to understand what the professor was asking for in the TODOs. But once I figured out all of those I was able to easily code the program.
2. Test code :

Player player;

player.setRound(4);

assert(player.getScore() == 0);//checks the intialization of a players score

player.setRound(3);

player.roll(3);

player.roll(2);

player.roll(1);

assert(player.getScore() == 1);//checks that the players score will only increment if the roll value is equal to the round

player.setRound(2);

assert(player.getScore() == 0);//checks to see that the player's score gets reset to 0 when .setround is called

player.setRound(4);

player.roll(3);

player.roll(2);

player.roll(1);

player.roll(4);

player.roll(4);

assert(player.getScore() == 2);//checks that the score can go above 2, and that for every mValue=mRound the score increases

player.setRound(5);

assert(player.getScore() == 0);//checks that the round gets reseet to 0 when the score >1

Board board;

board.setCurrentRound(4);

board.markHumanAsWinner();

assert(board.countUpComputerRoundWins() == 0);//checks to see that when no computer wins are there .countComputerRoundWins=0

assert(board.countUpHumanRoundWins() == 1);//checks to see that when .markHumanAsWinner is called .countUpHumanRoundWins goes up by 1; checks functionality

board.setCurrentRound(3);//

board.markComputerAsWinner();

assert(board.countUpComputerRoundWins() == 1);

assert(board.countUpHumanRoundWins() == 1);

//checks functionality for countUpComputerRoundWins and markComputerAsWinner

//HumanRoundWins stays the same

Board board2;

board2.setCurrentRound(4);//checks that you can jump around array elements non-consecutively

board2.markHumanAsWinner();

board2.setCurrentRound(3);

board2.markComputerAsWinner();

board2.setCurrentRound(2);

board2.markComputerAsWinner();

board2.setCurrentRound(1);

board2.markComputerAsWinner();

board2.markHumanAsWinner();

assert(board2.countUpComputerRoundWins() == 3);//checks counting of multiple computer and human wins that are non-consectutive

assert(board2.countUpHumanRoundWins() == 2);

Board board3;

board3.setCurrentRound(4);//checks that you can jump around array elements non-consecutively

board3.markHumanAsWinner();

board3.setCurrentRound(3);

board3.markHumanAsWinner();

board3.setCurrentRound(2);

board3.markHumanAsWinner();

board3.setCurrentRound(1);

board3.markHumanAsWinner();

board3.setCurrentRound(5);

board3.markHumanAsWinner();

board3.setCurrentRound(6);

board3.markHumanAsWinner();

assert(board3.countUpComputerRoundWins() == 0);

assert(board3.countUpHumanRoundWins() == 6);//when the entire array is full of wins accurately counts them

Bunco game;

Die d2;

d2.setValue(2);

game.nextRound();//sets mRound=2

game.humanPlay(d2);

game.computerPlay(d2);

game.endTurn();//new round, when called increases mRound -->3

assert(game.determineRoundOutcome() == Bunco::ROUNDOUTCOME::NOTDECIDED);//score is tied so this round is not determined yet

assert(game.determineGameOutcome() == Bunco::GAMEOUTCOME::GAMENOTOVER);//checking the function of determineGameOutcome

assert(game.getHuman().getScore() == 1);//checking that the Human score does not reset when no winner is determined

assert(game.getComputer().getScore() == 1);//checking that the computer score does not reset when no winner is determined

assert(game.getBoard().countUpHumanRoundWins() == 0);//checking that no wins are assigned when no winner is determined

assert(game.getBoard().countUpComputerRoundWins() == 0);

Bunco game1;

Die d3;

d3.setValue(2); Die d4;

d4.setValue(3);

game1.nextRound();

game1.nextRound();

game1.humanPlay(d3);

game1.computerPlay(d3);

game1.endTurn();

game1.humanPlay(d4);

game1.computerPlay(d4);

game1.endTurn();

game1.humanPlay(d3);

game1.computerPlay(d4);

game1.endTurn();

assert(game1.determineRoundOutcome() == Bunco::ROUNDOUTCOME::COMPUTERWON);//checks that the function determineRoundOutcome

assert(game1.determineGameOutcome() == Bunco::GAMEOUTCOME::GAMENOTOVER);

assert(game1.getHuman().getScore() == 1);//checks that scores do not get reset even when a winner is determined

assert(game1.getComputer().getScore() == 2);

assert(game1.getBoard().countUpHumanRoundWins() == 0);

assert(game1.getBoard().countUpComputerRoundWins() == 1);//checks that endturn will mark computer as winner whenever computer's score is larger than human score

Bunco game3;

d2.setValue(2);

d3.setValue(3);

d4.setValue(4);

game3.nextRound();

game3.nextRound();

game3.nextRound();

game3.humanPlay(d2);

game3.computerPlay(d2);

game3.endTurn();

game3.humanPlay(d3);

game3.computerPlay(d3);

game3.endTurn();

game3.humanPlay(d4);

game3.computerPlay(d3);

game3.endTurn();

assert(game3.determineRoundOutcome() == Bunco::ROUNDOUTCOME::HUMANWON);

assert(game3.determineGameOutcome() == Bunco::GAMEOUTCOME::GAMENOTOVER);

assert(game3.getHuman().getScore() == 1);

assert(game3.getComputer().getScore() == 0);

assert(game3.getBoard().countUpHumanRoundWins() == 1);//checks that endturn will mark human as winner whenever humans score is larger than the computer score

assert(game3.getBoard().countUpComputerRoundWins() == 0);

Bunco game4;

Die d1;

d1.setValue(1);

d2.setValue(2);

d3.setValue(3);

d4.setValue(4); Die d5;

d5.setValue(5);

Die d6;

d6.setValue(6);

game4.humanPlay(d1);

game4.computerPlay(d2);

game4.endTurn();

game4.nextRound();

game4.humanPlay(d1);

game4.computerPlay(d2);

game4.endTurn();

//only nextRound can start the next round and reset the scores to 0

assert(game4.getHuman().getScore() == 0);

assert(game4.getComputer().getScore() == 1);

game4.nextRound();

assert(game4.getHuman().getScore() == 0);

assert(game4.getComputer().getScore() == 0);

game4.humanPlay(d3);

game4.computerPlay(d2);

game4.endTurn();

game4.nextRound();

game4.humanPlay(d1);

game4.computerPlay(d4);

game4.endTurn();

game4.nextRound();

game4.humanPlay(d5);

game4.computerPlay(d2);

game4.endTurn();

game4.nextRound();

game4.humanPlay(d1);

game4.computerPlay(d6);

game4.endTurn();

assert(game4.determineRoundOutcome() == Bunco::ROUNDOUTCOME::COMPUTERWON);

assert(game4.determineGameOutcome() == Bunco::GAMEOUTCOME::TIEDGAME);//checks the gameoutcome function when the score is tied and 6 games have passed

assert(game4.getHuman().getScore() == 0);

assert(game4.getComputer().getScore() == 1);//only next round can reset the score to 0

assert(game4.getBoard().countUpHumanRoundWins() == 3);

assert(game4.getBoard().countUpComputerRoundWins() == 3);

Bunco game5;

d1.setValue(1);

d2.setValue(2);

d3.setValue(3);

d4.setValue(4);

d5.setValue(5);

d6.setValue(6);

game5.humanPlay(d1);

game5.computerPlay(d2);

game5.endTurn();

game5.nextRound();

game5.humanPlay(d1);

game5.computerPlay(d2);

game5.endTurn();

game5.nextRound();

game5.humanPlay(d2);

game5.computerPlay(d3);

game5.endTurn();

game5.nextRound();

game5.humanPlay(d1);

game5.computerPlay(d4);

game5.endTurn();

game5.nextRound();

game5.humanPlay(d5);

game5.computerPlay(d2);

game5.endTurn();

game5.nextRound();

game5.humanPlay(d1);

game5.computerPlay(d6);

game5.endTurn();

assert(game5.determineRoundOutcome() == Bunco::ROUNDOUTCOME::COMPUTERWON);

assert(game5.determineGameOutcome() == Bunco::GAMEOUTCOME::COMPUTERWONGAME);//checks the gameoutcome function when the computer wins and 6 games have passed

assert(game5.getBoard().countUpHumanRoundWins() == 2);

assert(game5.getBoard().countUpComputerRoundWins() == 4);

Bunco game6;

d1.setValue(1);

d2.setValue(2);

d3.setValue(3);

d4.setValue(4);

d5.setValue(5);

d6.setValue(6);

game6.humanPlay(d1);

game6.computerPlay(d2);

game6.endTurn();

game6.nextRound();

game6.humanPlay(d2);

game6.computerPlay(d1);

game6.endTurn();

game6.nextRound();

game6.humanPlay(d2);

game6.computerPlay(d3);

game6.endTurn();

game6.nextRound();

game6.humanPlay(d4);

game6.computerPlay(d3);

game6.endTurn();

game6.nextRound();

game6.humanPlay(d5);

game6.computerPlay(d2);

game6.endTurn();

game6.nextRound();

game6.humanPlay(d6);

game6.computerPlay(d1);

game6.endTurn();

assert(game6.determineRoundOutcome() == Bunco::ROUNDOUTCOME::HUMANWON);

assert(game6.determineGameOutcome() == Bunco::GAMEOUTCOME::HUMANWONGAME);//checks the game outcome function when the human won more games than the computer after 6 games have passed

assert(game6.getBoard().countUpHumanRoundWins() == 5);

assert(game6.getBoard().countUpComputerRoundWins() == 1);

//Stahl test code

Die d;

for (int i = 1; i <= 100; i++)

{

d.roll();//calls .roll() 100 times

int value = d.getValue();

assert(value >= 1 && value <= 6);//tests that each .roll() will always be a value between 1 and 6

}

Player p;

assert(p.getScore() == 0);//when a player is creted their score is initialized to 0

p.setRound(1);//sets mRound=1

assert(p.roll(6) == 6);//checks that the cheat part of player works

assert(p.getScore() == 0);//score isnt changed when roll happens

assert(p.roll(1) == 1);

assert(p.getScore() == 1);//score only increases when .roll()=mRound, checks that mRound is defaulted to 1 when player is created

p.setRound(6);

assert(p.getScore() == 0);//tests when setround is called to reset the score to zero,

assert(p.roll(6) == 6);

assert(p.getScore() == 1); //tests that the setround works properly and makes mRound=6

//since mRound=6 and .roll=6, score increases by 1

Die d11;

d11.setValue(1);

Die d21;

d21.setValue(2);

Die d31;

d31.setValue(3);

Die d41;

d41.setValue(4);

Die d51;

d51.setValue(5);

Die d61;

d61.setValue(6);

Bunco b;//round is automatically set to 1

assert(b.determineRoundOutcome() == Bunco::NOTDECIDED);//checks when round hasnt been played yet to put round outcome as not decided

b.computerPlay(d51);

b.humanPlay(d51);

b.endTurn();

assert(b.determineRoundOutcome() == Bunco::NOTDECIDED);//when no one rolls a value that equals the mRound the round is not decided

b.computerPlay(d11);

b.humanPlay(d11);

b.endTurn();

assert(b.determineRoundOutcome() == Bunco::NOTDECIDED);//when both players roll a value that equals the mRound the round is not decided

b.computerPlay(d11);

b.humanPlay(d21);

b.endTurn();

assert(b.determineRoundOutcome() == Bunco::COMPUTERWON);//when one player rolls a value that equals the mRound the round is decided in that players favor

b.nextRound();

Player foo;

foo.setRound(2);

int counter = 0;//tests the random function of player; ensures that the .roll() is increasing the score as it needs to

int amount = foo.roll();

if (amount == 2) {

counter++;

}

assert(foo.getScore() == counter);

Bunco foo1;

Player z;

foo1.nextRound();

int counter1 = 0;//tests random function of humanplay and ensures increasing score as player gets the correct roll

for (int i = 1; i < 100; i++) {

int amount1 = foo1.humanPlay();

if (amount1 == 2) {

counter1++;

}

z = foo1.getHuman();

assert(z.getScore() == counter1);

}

Bunco foo2;

Player ze;

foo2.nextRound();

int counter10 = 0;//tests random function of computerplay and ensures increasing score as player gets the correct roll

for (int i = 1; i < 100; i++) {

int amount1 = foo2.computerPlay();

if (amount1 == 2) {

counter10++;

}

ze = foo2.getComputer();

assert(ze.getScore() == counter10);

}